CI102

LAB 7

Objectives

1. Identify the entities of a detailed design for your project.
2. Create detailed designs for a subset of the entities of your project.
3. Assess your team's technical capability compared to the technical needs of the project.

Designers need to specify the details of the entities that make up the system. These definitions should be sufficiently detailed that the design can be given to a developer and the developer can create the entity as envisioned by the designer.

Once your team starts to develop a design, you should also be developing a better understanding of the technologies and skill levels needed to build the product. As a separate task, this lab will also provide a chance for you to assess your team's capability to work on the project and identify learning or skill development you may need.

Procedure

Step 1 – Draft a list of entities for your project

You should consider the following types of entities:

- Screens (or Web pages)
- Database tables
- Files (e.g., data that is stored as part of the system but not stored in a database)
- Code (modules, objects, or functions)

Use Figure 7-1 to list all the system entities that you can identify. A good way to start is to pick one area and focus on that. For example, if your system has a significant user interface, start by trying to name all the screens that would comprise your interface. For each entity you list:

- Enter a type, e.g., "screen"
- Give it a meaningful name, e.g., "CustomerProfile"
- Provide any short notes or explanation needed to identify the screen, e.g., "This screen captures customer information and preferences."

Step 2 – Create detailed designs for at least 4 of your entities.

You will not be able to design all the entities of your system in this lab, but this step will get you started. Pick 4 entities that you think you understand the best at this point, and create a design for them. Every entity should have a name, type, and design details. Templates are provided to help you create detailed design for screens, database tables, and code functions.

Step 3 – Review your detailed designs.

After creating your designs, review them for completeness and clarity. Ask yourself this question: "If I was the developer and a designer handed me this design, would I know what to build without needing to ask a lot of questions?

If you have created the design entities as a team, set them aside for a few minutes before review each one. If you have worked in sub-groups within your team to create the designs, then exchange designs so the reviewer is a different person than the creator of a design.

Revise your designs based on the review.

Step 3 – Assess your team's capability to complete this project.

Once you have an architectural overview and the beginning of a design, you should be able to assess capability and identify things that someone on the team may need to learn. Use Figure 7-5 to summarize this information.

3.A – List the technologies you need for your project using the column on the left. Consider things such as programming languages, operating systems, specialized data sources, software libraries, support tools, and hardware.

3.B – List each team member at the top of a column, and then evaluate that person's knowledge of the technology in each row. For the column for each team member, use the following values:

1 – No knowledge or not much relative to the needs of this project
2 – Enough knowledge to accomplish part but not all of this project
3 – Knowledge probably sufficient for this project

3.C – Discuss within your team how you will start to gain capabilities that you are missing. You do not need to turn in results of this discussion in this lab, but will need to address this in the coming weeks.

What to Turn In

In order to obtain full credit for this lab, ***each team*** must turn in:

1. Figure 7-1 – Possible System Entities
2. Detailed designs for at least 4 entities in your system. Use the templates in Figures 7-2 through 7-4 to get started.
3. Figure 7-5 – Team Capability Assessment

Figure 7-1 – Possible System Entities

Product: Social Hour
Team: 85
Date: 2/24/17

Type	Name	Description or Notes
screen	Main Menu	Displays live feed
screen	Add/Edit Event	Allows user to add or edit events with field name and field values
screen	Settings	User can manage application and profile
screen	Calendar	Allows user to view calendar by the day, week, or month
screen	Friends	Allows user to view friend list and add friends
screen	Group	Allows user to create, view, and join groups
screen	***Notification***	***Notifies user when invited to event or group***
database	~~MongoDB~~ ***Google Firebase***	~~User data will be stored, MongoDB will be interfaced with RoboMongo~~ ***User data will be stored through Google Authentication, and the rest of the data will be stored through Google Firebase***
database	Google	User calendar would be stored using Google Calendar
files	Cache	
function	create_group	Function that creates an initially empty group with a specific set of attributes
function	establish_friendship	Function that establishes a friendship between two people objects
technologies	Android Studio	used to program the application
technologies	GNU Image Manipulation Program (GIMP)	used to design the user interface and the logo of the application
technologies	Inkscape	used to design the icon and vector manipulation
files	***Event Data***	***Organized through a Java class, includes arrays and strings regarding event data***
technologies	***Atlassian Bitbucket***	***Used as a revision control system for our Android Studio files, our laboratory deliverables, and various assets regarding the program***

CI102

ENTITY 1: (portions of this page have been changed for CI 103)
Type: Screen

Name: Main menu screen

Purpose: This screen serves as the home page for the application, meeting requirements 3, 4, and 5, which are the mobile interface, social networking, and stationary header for the application.

Description: Figure 1 shows the layout for this screen. This screen allows users to scroll down their feed of group and friend activity and access the other features of the application

The screen contains the following elements:
Calendar button: This element will allow the user to click the Calendar button to access the calendar.
Menu Bar: This menu bar will allow us to swap between three of the major pages: The dashboard (feed), the Groups page, and the Friends page.

Layout:

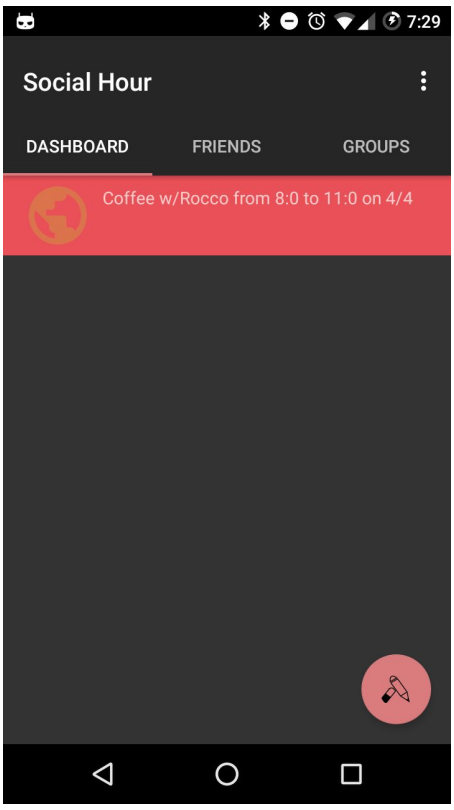

Figure 1: Main Screen (not pictured: "Social Hour" text being replaced with "Calendar" and centered towards the middle"

ENTITY 2:
Name: establish_friendship

Type: Function

Purpose: This function is needed to meet requirement 4, which is the social networking portion of the application.

Parameters: The following parameters are used to call this function:

Name	Data Type	Notes
friend1	friend object	passed by reference
friend 2	friend object	passed by reference

Return Type: boolean - returns true if the friendship connection succeeded, returns false if failed

Processing:

```
pseudocode:
bool establish_friendship(friend &friend1, friend &friend2)
     if(!friend1.is_friends_with(friend2))
          friend1.friends_list.push_back(friend2)
          friend2.friends_list.push_back(friend1)
          return true
     else
          return false
```

ENTITY 3
Name: create_event

Type: Function

Purpose: This function is needed to meet requirement 4, which is the event creation and social media portion of the design.

Parameters: The following parameters are used to call this function:

Name	Data Type	Notes
event_name	string	
is_all_day	boolean	
event_time	integer	empty of is_all_day is true
group_affiliation	group	Potentially can be empty
event_date	string	
friends_invited	vector<&friends>	Potentially can be empty individual friends passed by reference if necessary
privacy	integer	0 if public, 1 if limited to friends, 2 if limited to best friends

Return Type: event object
Processing:

```
pseudocode:
bool create_event(string event_name, bool is_all_day, int event_time,
group
                  group_affiliation, string event_date, vector<&friends>
                  friends_invited, integer privacy)
     if invalidDate(event_name) throw exception //cancel event creation
     if invalidTime(event_time) throw exception; //cancel event creation
     return event(event_name, is_all_day, event_time, group_affiliation,
                  event_date, friends_invited, privacy) //return the event
                                                        //for use in java
                                                        //code
```

CI102

ENTITY 4:

Type: Screen

Name: Create Menu screen

Purpose: This screen is needed for requirement 3, the mobile interface of Social hour.

Description: Figure 4 1 shows the layout for this screen. This screen allows users to edit their preferences for the event before creating the event, partially using the create_event function.

The screen contains the following elements:

Calendar button: This element will allow the user to click the calendar button to access the calendar.
Menu bar: This element allows users to switch between the five main views : Friends, Groups, Main Page, Add/Edit Event, and Settings, in that order.
Event date: number box (textbox with restrictions to integers) that allows user to put event date in the application.
Is all day: checkbox for the user to indicate if the event takes place all day.
Event time: number box that allows users to input the time of the event. Input auto-formats (including placing the colon in the middle). Greys out if "Is all day" is checked.
Add Group: textbox that is actually a button, shows dropdown of groups the member is a part of. When added, the group is inserted into the textbox as a string.
Add Friends textbox that is actually a button, shows dropdown of the member's friends. When added, the friend is inserted into the textbox as a string.

Layout:

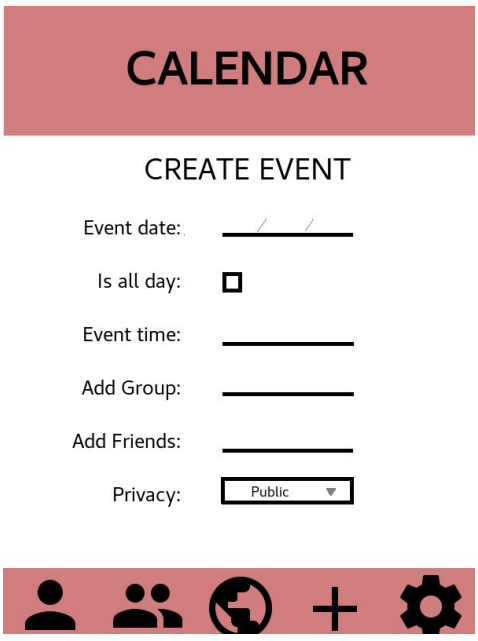

Figure 4-1: Create Event Screen

CI102

ENTITY 5 (This entire entity is added for CI 103):

Type: Screen

Name: Friends Menu Screen

Purpose: This screen is needed for requirement 3, the mobile interface of Social hour.

Description: Figure 5-1 shows the layout for this screen. This screen allows users to view all of the friends they have made through the social network, and manage friends by selecting specific friends.

The screen contains the following elements:

Calendar button: This element will allow the user to click the Calendar button to access the calendar.
Menu Bar: This menu bar will allow us to swap between three of the major pages: The dashboard (feed), the Groups page, and the Friends page.
Add Event button: floating button that allows the user to add an event
Friends list: list of the current friends a user has - user testing will reveal whether or not we actuall want to display user icons

Layout:

Figure 5-1: Create Event Screen (NOT PICTURED: "Social Hour" text replaced with "Calendar" and placed in the center

ENTITY 6: (This entire entity is new to CI103)
Name: EventItem

Type: System Entity (Java Object, Data)

Purpose: This function is needed to meet requirements 3 4, which is the social networking portion of the application and the mobile interface of the application. All of these variables have accessors and mutators using void and data type functions.

Data Structure:

Type	Name	Notes
private int	start_hour	Starting hour of the event stored as an int
private int	end_hour	Ending hour of the event stored as an int
private int	start_minute	Starting minute of the event stored as an int
private int	end_minute	Ending minute of the event stored as an int
private String	event_title	Title of the event stored as a string
private String	event_description	Description of the event stored as a string
private int	dayOfMonth	Date in the month of the event stored as an int
private int	monthOfYear	Month in the year of the event stored as an int
private int	year	Year of the event stored as an int
private boolean	isAllDay	Boolean flag determining whether or not the event consumes the whole day - code functions will ignore int values start_hour. end_hour, start_minute, and end_minute if this is checked.
constructor	EventItem()	Constructs an EventItem object and initializes all of the data values.

Implementation of this object is subject to change as research in Firebase, user testing, and other project-related activities occur.

ENTITY 7: (This entire entity is new to CI103)
Name: public void finish()

Type: void function

Purpose: This function is needed to meet requirements 3 4, which is the social networking portion of the application and the mobile interface of the application. This function completes the Create Event function (add_menu_activity in Android Studio); the superclass catches all of the data using the data in the putExtra() function and uses it to call add_event.

There are no return variables for this function, because the entire Java activity is passed to the superclass.

Sample code for the current design of finish():

```
public void finish() {
   Intent data = new Intent();


   edit_event_name_textedit = (TextView) findViewById(R.id.edit_event_name_edittext);
   edit_event_description = (TextView) findViewById(R.id.event_description_textbox);
   final CheckBox is_all_day_checkbox = (CheckBox) findViewById(R.id.is_all_day_checkbox);
   boolean isAllDay = is_all_day_checkbox.isChecked();

   String event_name = edit_event_name_textedit.getText().toString();
   String event_description = edit_event_description.getText().toString();

       //extra values can be placed in here as code develops
   data.putExtra("event_year", this.event_year);
   data.putExtra("event_month", this.event_month);
   data.putExtra("event_day", this.event_day);
   data.putExtra("event_start_hour", this.start_hour);
   data.putExtra("event_end_hour", this.end_hour);
   data.putExtra("event_start_minute", this.start_minute);
   data.putExtra("event_end_minute", this.end_minute);
   data.putExtra("event_name", event_name);
   data.putExtra("event_description", event_description);
   data.putExtra("is_all_day", isAllDay);

   setResult(RESULT_OK, data);
   super.finish();
}
```

Implementation of this object is subject to change as research in Firebase, user testing, and other project-related activities occur.

ENTITY 8: (This entire entity is new to CI103)
Name: private void handleSignInResult(GoogleSignInResult result)

Type: void function

Purpose: This function is needed to meet requirements 3 4, which is the social networking portion of the application and the mobile interface of the application. This function executes when the user successfully signs in to their google account, and adds all of the necessary values to the user profile so the application can use their data.

There are no return variables for this function. The only parameter is the data GoogleSignIn gives to the function.

Sample code for the current design of handleSignInResult(GoogleSignInResult result):

```
private void handleSignInResult(GoogleSignInResult result)
{
   Log.d(TAG, "handleSignInResult:" + result.isSuccess());
   if (result.isSuccess())//signed in successfully, shot authenticated UI
   {
       GoogleSignInAccount acct = result.getSignInAccount();
       //mStatusTextView.setText(getString(R.string.signed_in_fmt, acct.getDisplayName()));
       UserData.set_user_first_name(acct.getDisplayName());
       UserData.set_user_last_name(acct.getFamilyName());
       UserData.set_user_email(acct.getEmail());
       UserData.set_user_id(acct.getId());
       try {
           UserData.set_user_bitmap(MediaStore.Images.Media.getBitmap
                   (this.getContentResolver(), acct.getPhotoUrl()));
       }
       catch (IOException e) {
           UserData.set_user_bitmap(null);
       }
       UserData.set_user_given_name(acct.getGivenName());
       updateUI(true);
   }
   else//signed out, unauthenticated UI
   {
       updateUI(false);
   }
}
```

Implementation of this object is subject to change as research in Firebase, user testing, and other project-related activities occur.

ENTITY 9: (This entire entity is new to CI103)

Name: `public void onTimeSet(TimePicker view, int hourOfDay, int minute)`

Type: void function

Purpose: This function is needed to meet requirements 3 4, which is the social networking portion of the application and the mobile interface of the application. This function executes when the user wants to input a time, for example when selecting a start or end time for an event.

There are no return variables for this function, the only thing it does is manipulate the object.

Sample code for the implementation of onTimeSet:

```
   public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
        String am_pm;
        start_hour = hourOfDay;
        start_minute = minute;
        int start_hour_12 = start_hour;
        if(start_hour_12 > 12)
        {
            start_hour_12 -= 12;
            if (start_hour_12 == 12) am_pm = "am";
            else am_pm = "PM";
        }
        else if(start_hour_12 == 12) am_pm = "PM";
        else am_pm = "AM";
        start_time_diag_button.setText(String.format(Locale.getDefault(), "%02d", start_hour_12) +
":"
                + String.format(Locale.getDefault(), "%02d", start_minute) + " " + am_pm);
   }
```

This code has potential user testing changes: Whether or not the user wants time to be displayed in 24 hour format will have a strong effect on this code.

Figure 7-5 – Team Capability Assessment

Capabilities	Michael	Rocco	Dylan	Gavin
Ability to Access and Manage the Server	3	3	3	3
Ability to Program the Android Application	3	2	3	2
Ability to design the UX	1	2	2	3
Ability to perform data analytics on the database content	2	2	2	3
Ability to lead the team in a substantial direction	3	2	2	2

** The table values represent an assessment of team member capabilities. The values are:

1 – No knowledge or not much relative to the needs of this project
2 – Enough knowledge to accomplish part but not all of this project
3 – Knowledge probably sufficient for this project